They're…

Functional!

Efficient!!

Persistent data structures!!!

@AnjanaVakil

!!Con 2016

Fall 2, 2015

Functional Programming rocks!

Immutability
rocks!

Nobody sits like this rock sits.

You rock, rock.

The rock just sits, and is.

You show us how to just sit here, and that's what we need.

-- I ❤ Huckabees (2004)

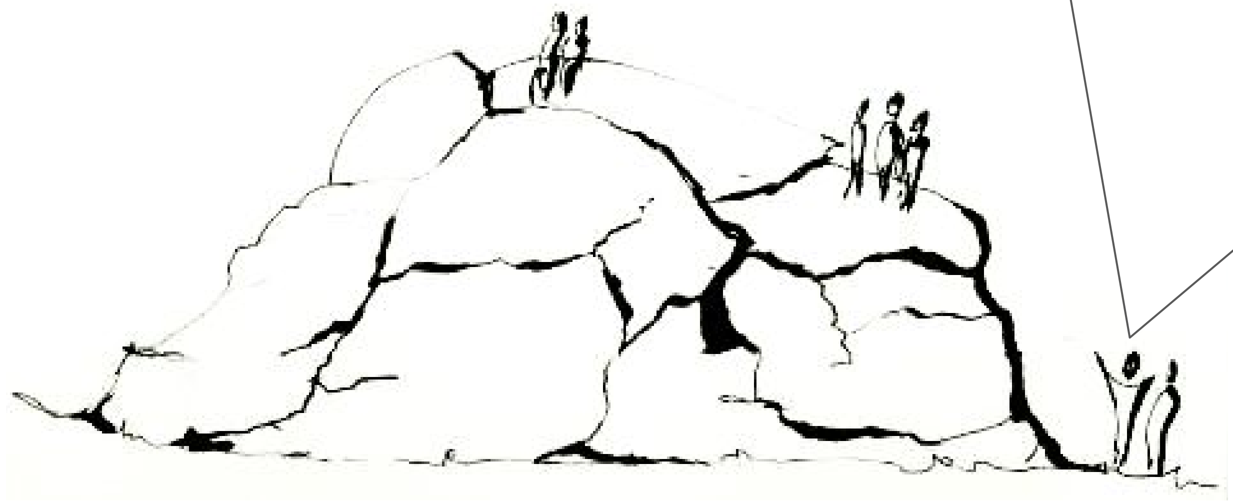
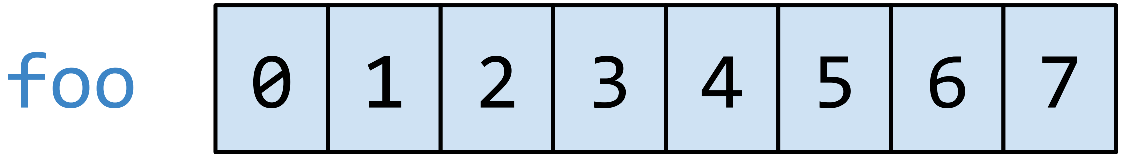
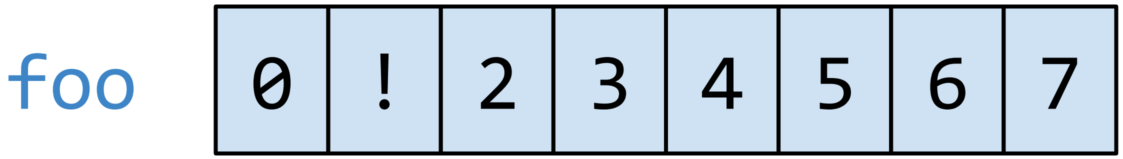


Illustration by Marco Piazza

In the land of mutability...

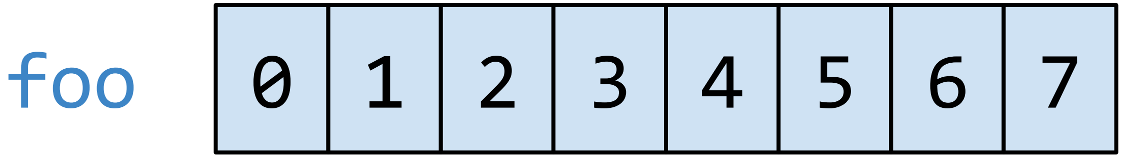


In the land of mutability...

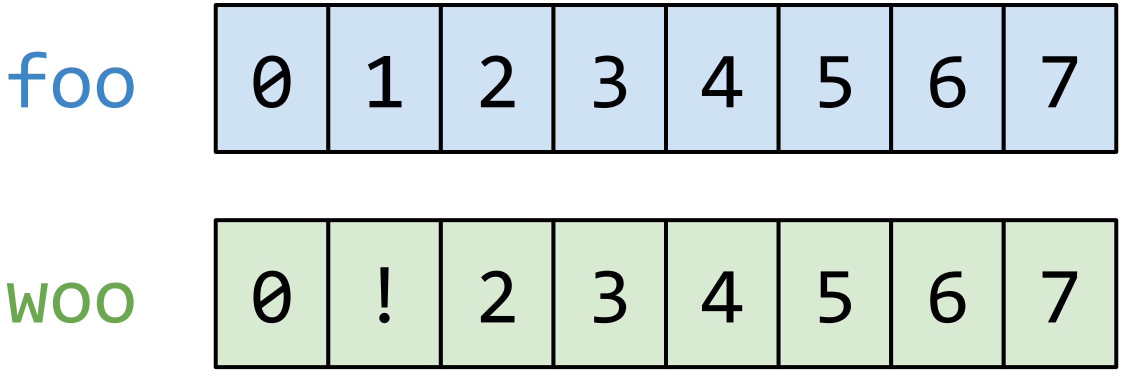


Overhead & bugs! :(

In the land of immutability...



In the land of immutability...



Copying wastes time/space! :(

There must be a better way...

Persistent Data Structures!

Old versions never change :)

(they just sit, and are)

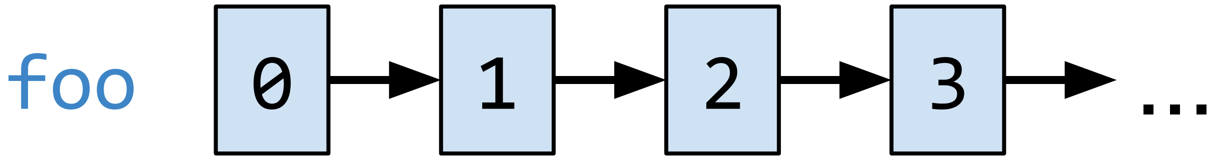
New versions created efficiently

:D

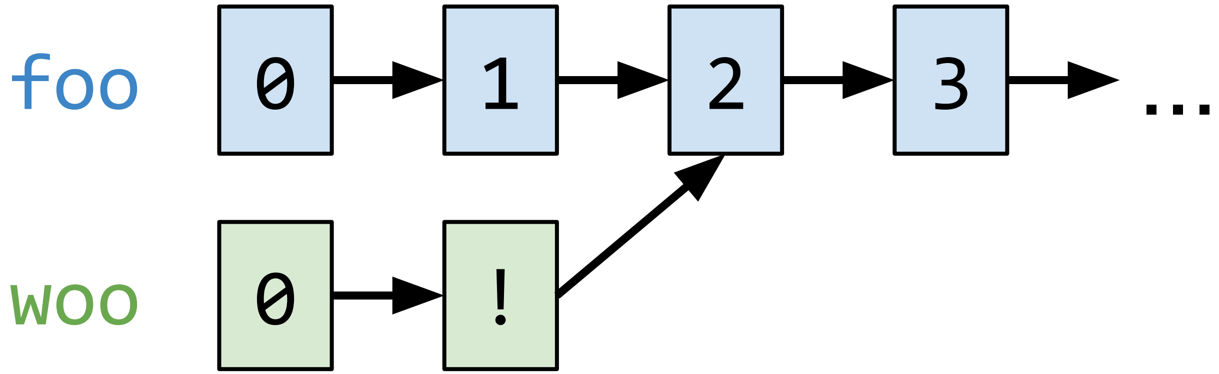
Magic?!

Reuse unchanged parts!

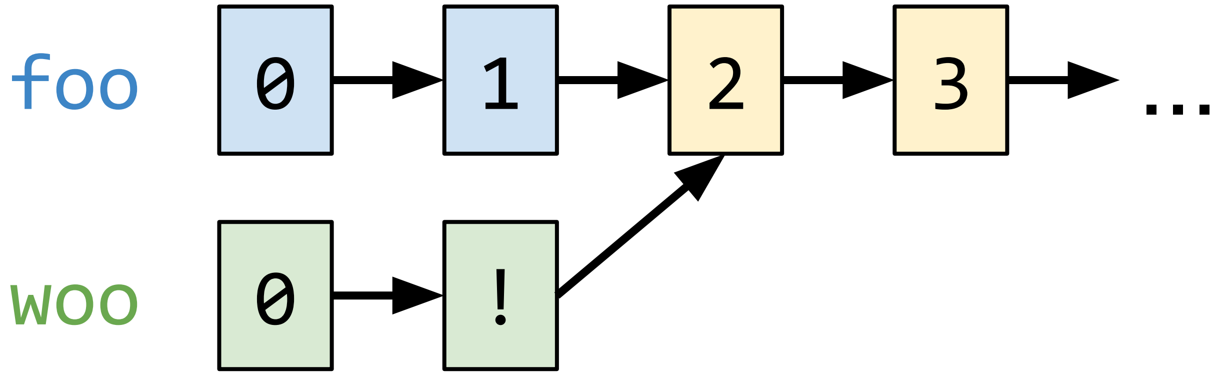
Linked lists!



Linked lists!



Linked lists!



Trees!

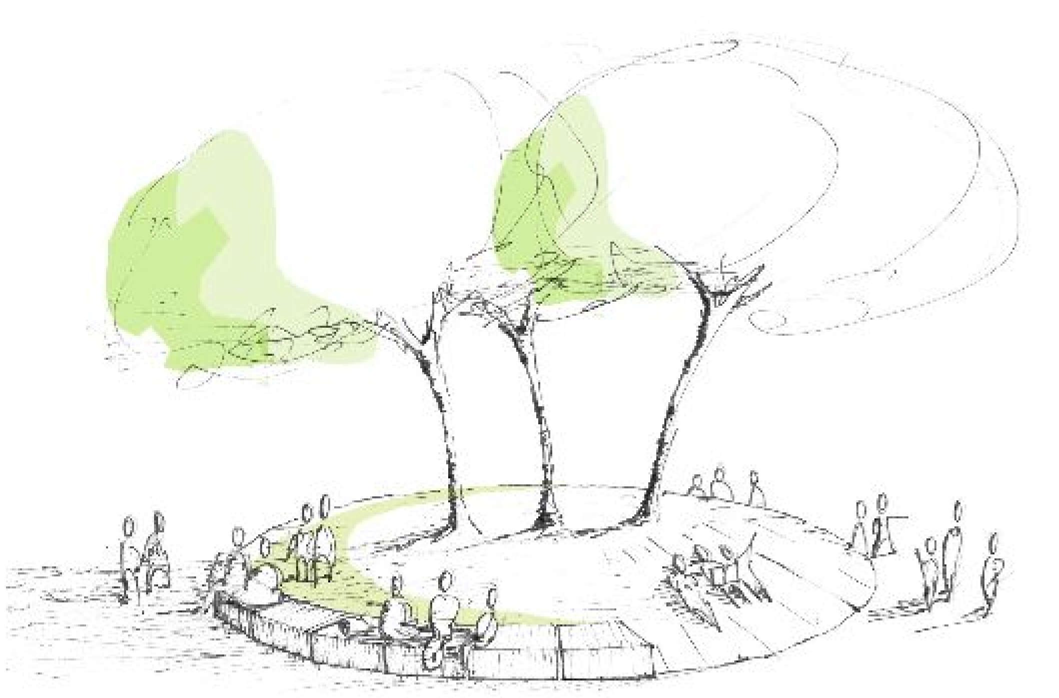
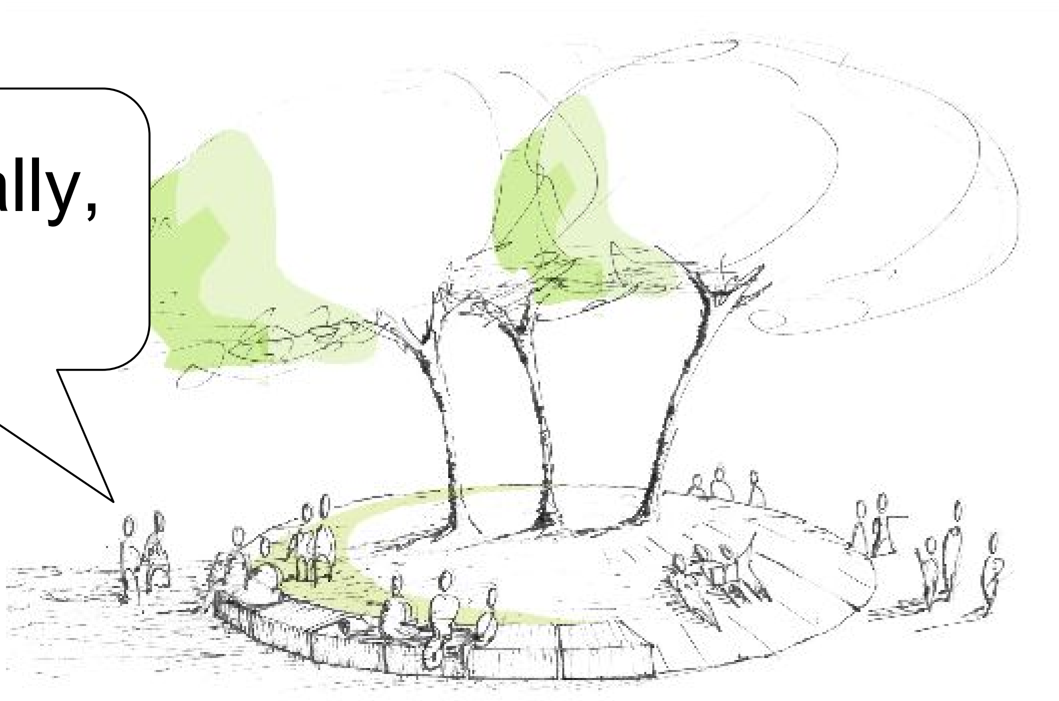


Illustration by Marco Piazza

Trees!



Well, actually, *tries*...

Illustration by Marco Piazza

Trees!

Illustration by Marco Piazza

Trees!

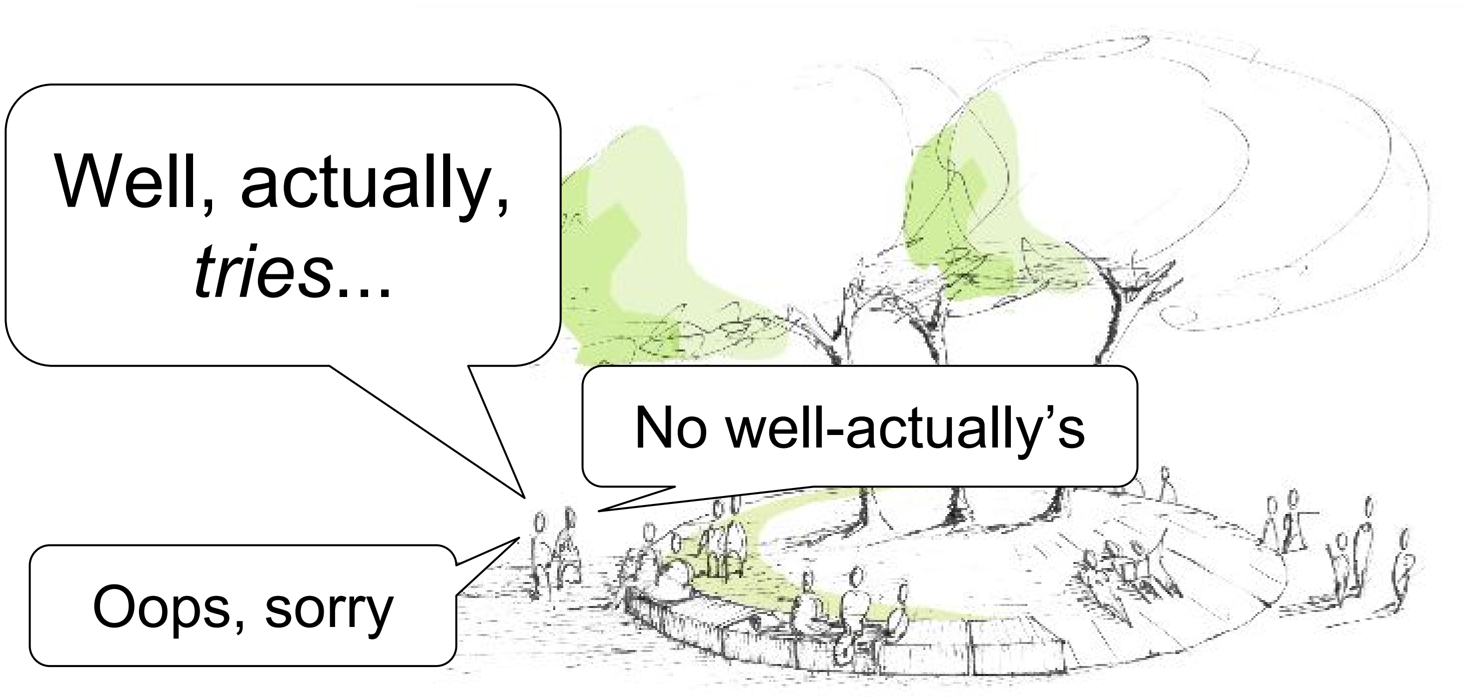
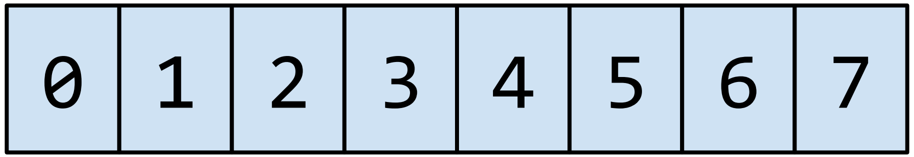
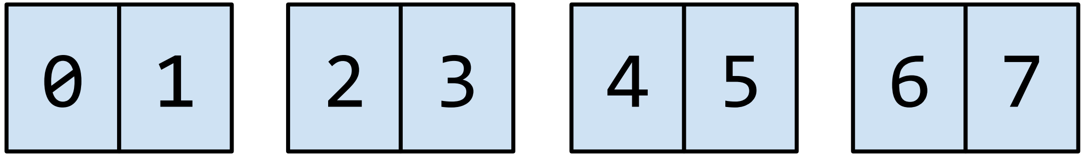


Illustration by Marco Piazza

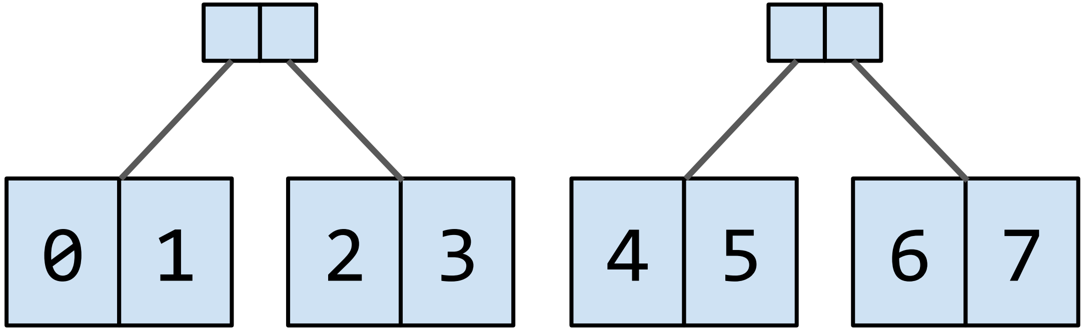
Trees!



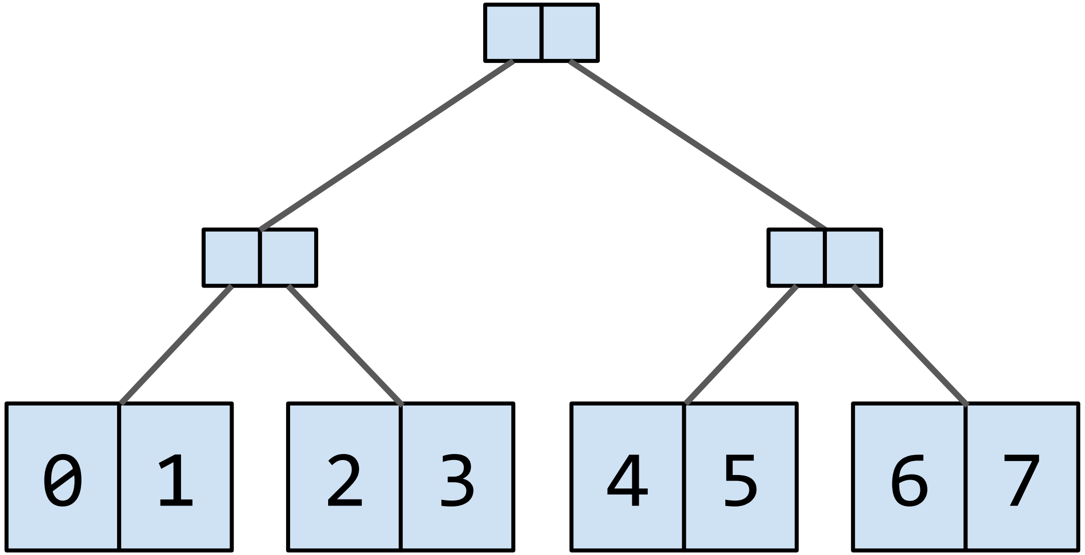
Trees!



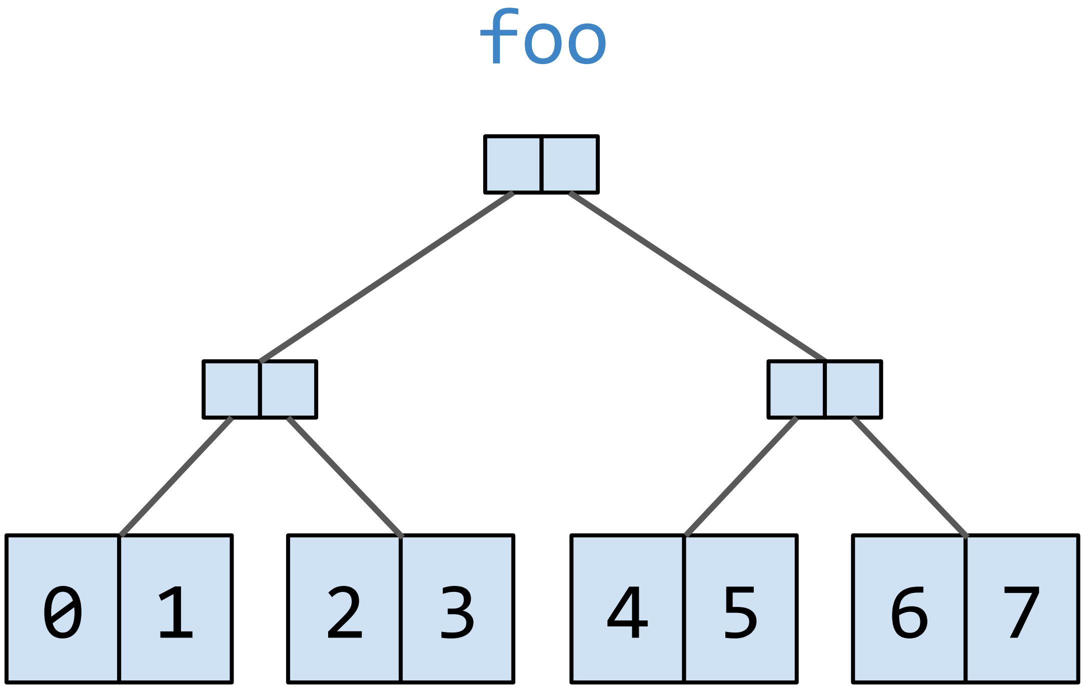
Trees!



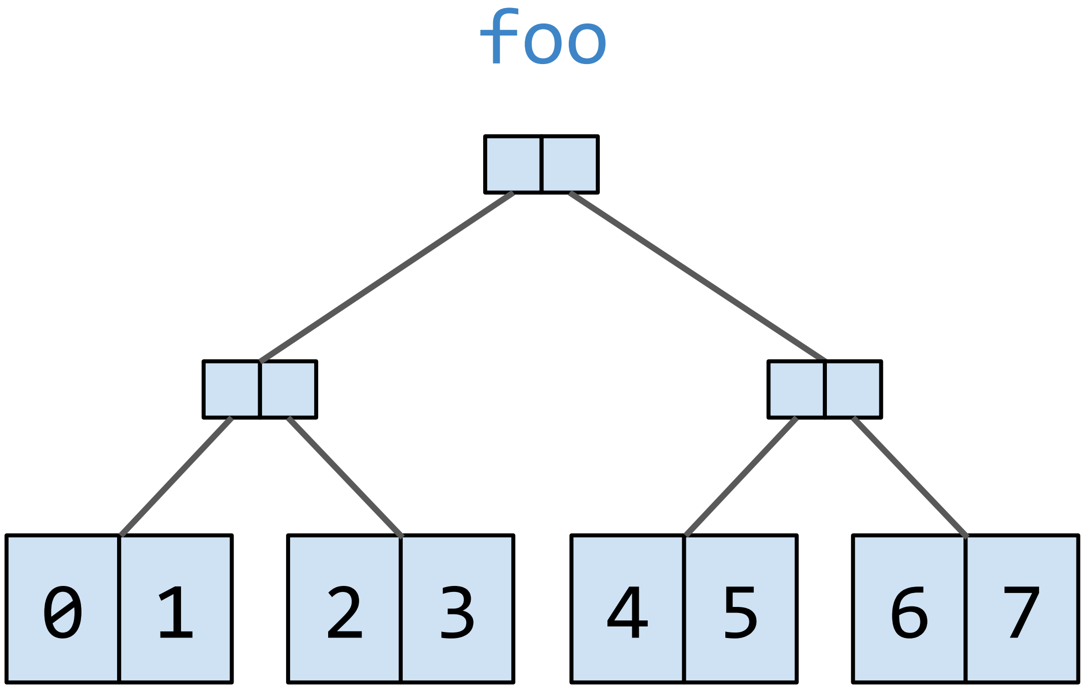
Trees!



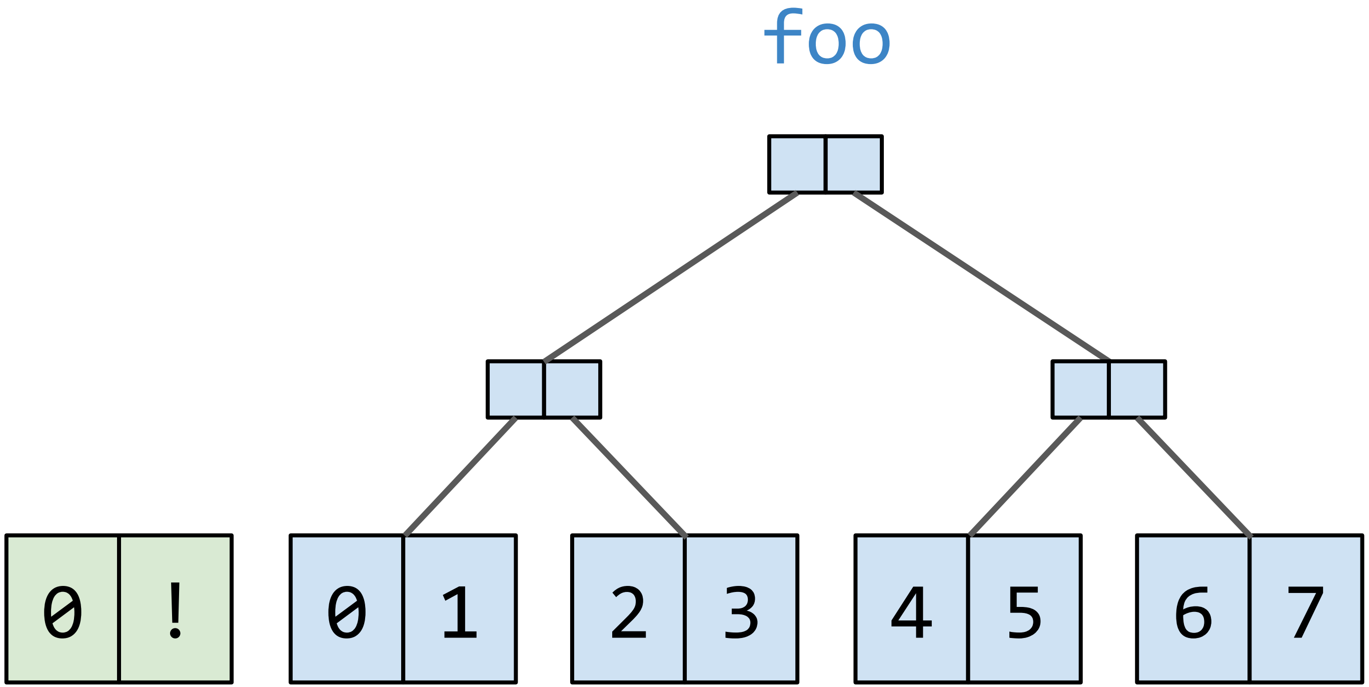
Trees!



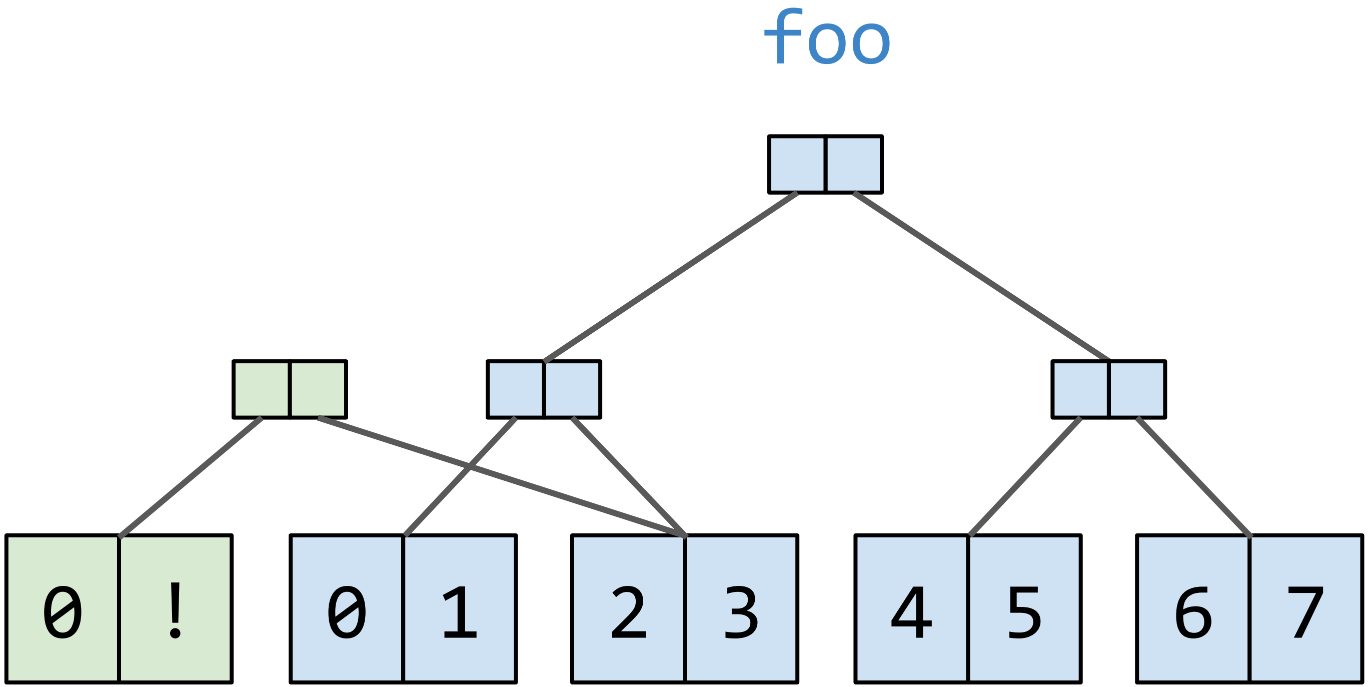
Trees! Path copying!!



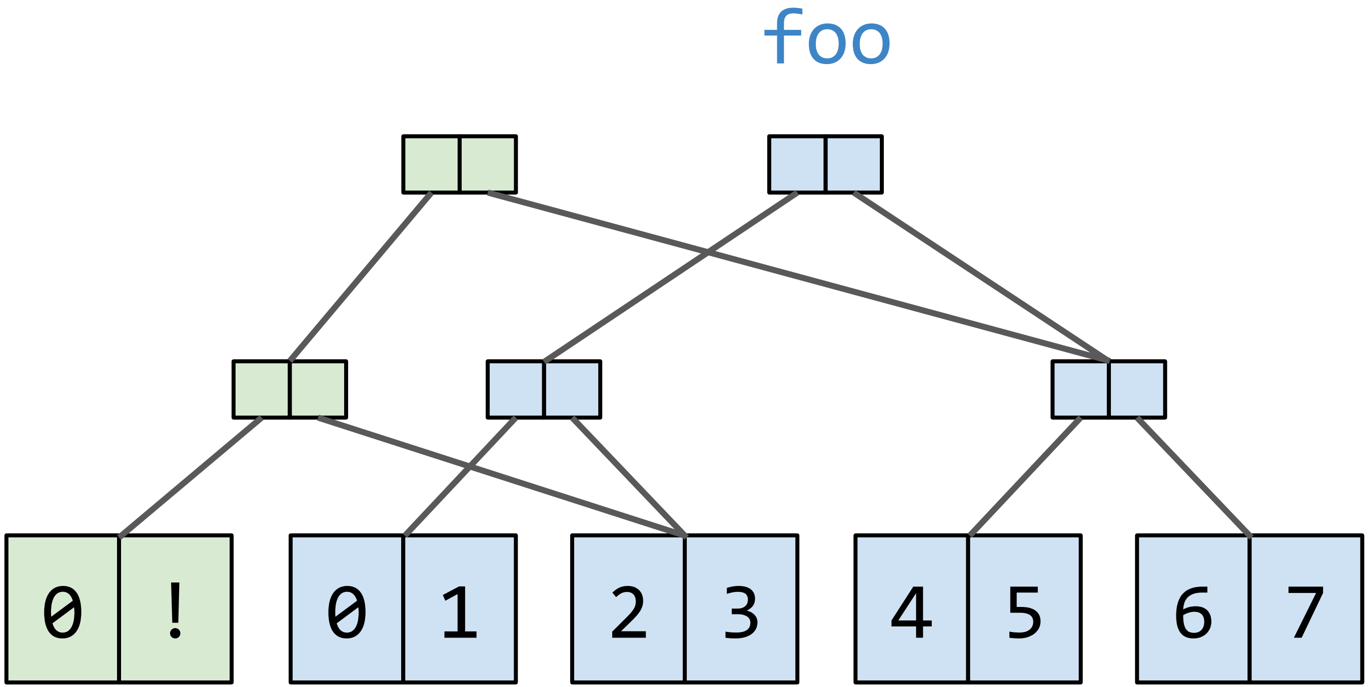
Trees! Path copying!!



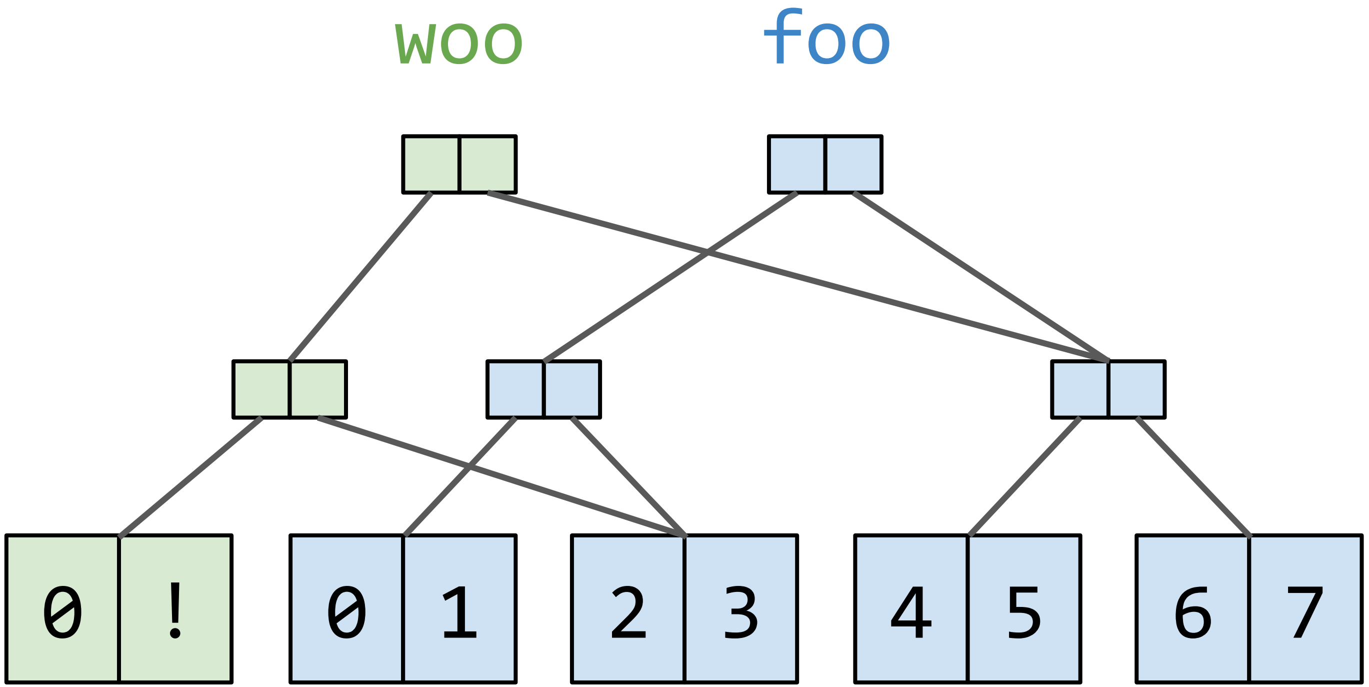
Trees! Path copying!!



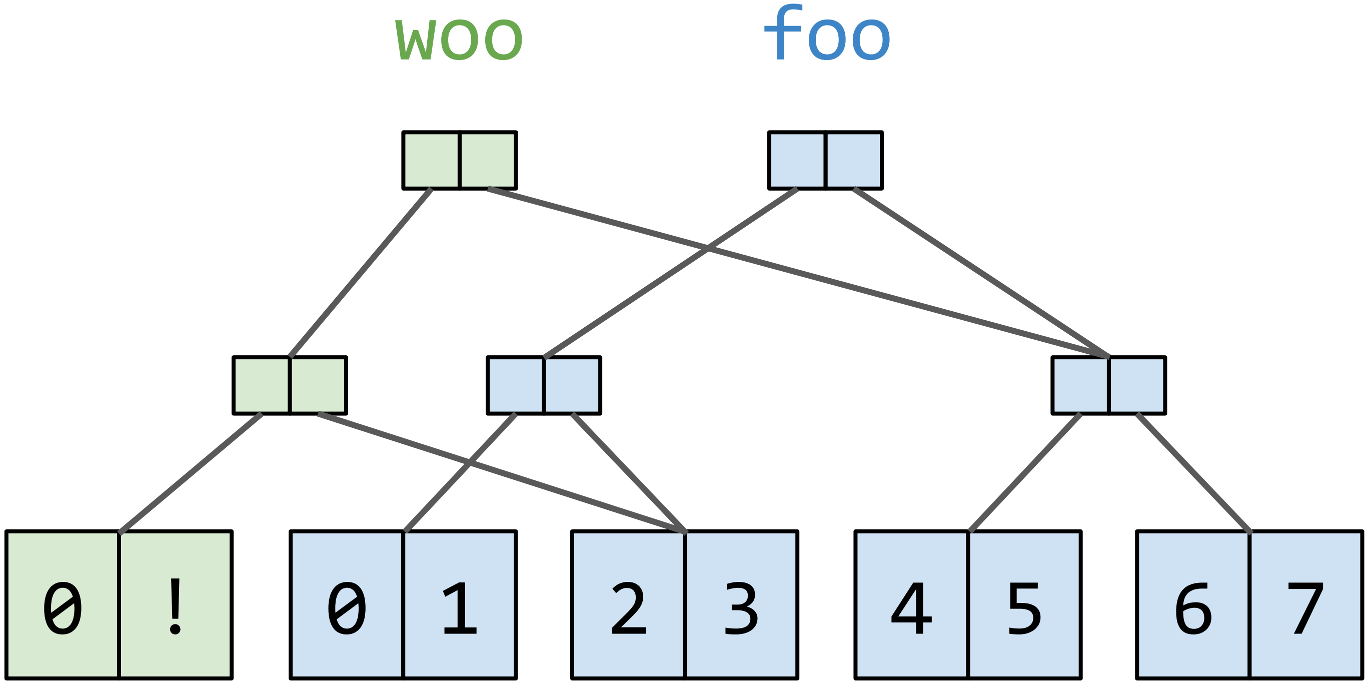
Trees! Path copying!!



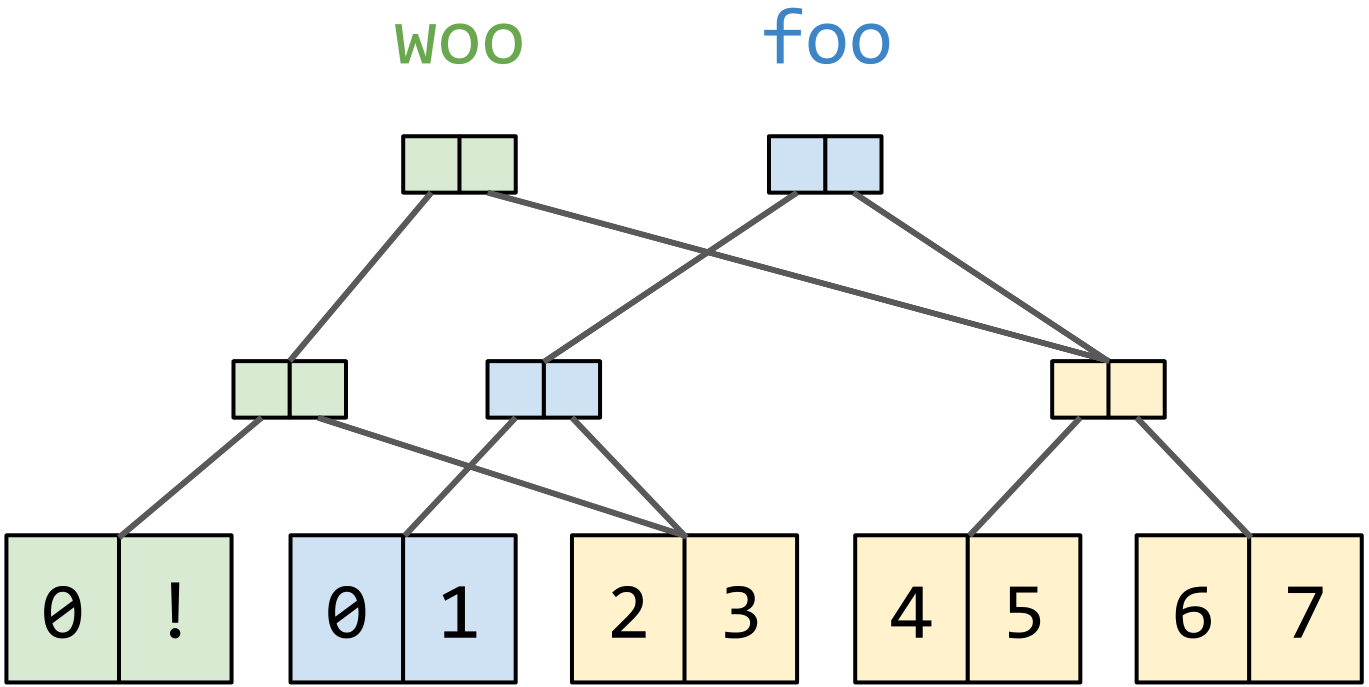
Trees! Path copying!!



Trees! Path copying!! Structural sharing!!!



Trees! Path copying!! Structural sharing!!!



Immutability == :)

Copying == :(

Sharing == :D

JavaScript!

Mori

https://swannodette.github.io/mori

```
var f = mori.vector(1,2);
var w = mori.conj(f, 3);
```

- ClojureScript port
- Functional API
- Fast

Immutable.js

https://facebook.github.io/immutable-js

```
var f = Immutable.List.of(1,2);
var w = f.push(3);
```

- JS through & through
- Public methods
- A bit smaller than Mori

Libraries for other languages too!

...or just use a functional language! :)
(say, Clojure)

Further Reading

“Understanding Clojure’s Persistent Vectors”
Jean Niklas L'orange
http://hypirion.com/musings/understanding-persistent-vector-pt-1

“Ideal Hash Trees”
Phil Bagwell
http://lampwww.epfl.ch/papers/idealhashtrees.pdf

Thanks for listening!

I’m @AnjanaVakil

Huge thanks to:

Recurse Center alums

Sal Becker (F2’15)

!!Con organizers